Laboratório 4 - Computação Concorrente

Ricardo Kaê - DRE 116 039 521

Considere abaixo, uma suposição do que seria o código montagem de um programa que dispara três threads $T1$, $T2$ e $T3$, com as funcionalidades descritas no *lab*4. Tal código foi feito para tornar mais claro, a ocorrência de determinado valor, na saída padrão.

```
                .data
format:        .ascii "%d\n"

                .text
                .global main

0 <T1>:
1         MOV x, %eax
2         SUB 1, %eax
3         MOV %eax, x               # x = x - 1
4         MOV x, %eax
5         ADD 1, %eax
6         MOV %eax, x               # x = x + 1
7         MOV x, %eax
8         SUB 1, %eax
9         MOV %eax, x               # x = x - 1
A         CMP x, -1                 # x == -1?
B         JE  D
C         CALL pthread_exit         # T1 finish
D         (...)                     # Prepara stack de printf
E         CALL printf

F <T2>:
10         MOV x, %eax
11         ADD 1, %eax
12         MOV %eax, x               # x = x + 1
13         MOV x, %eax
14         SUB 1, %eax
15         MOV %eax, x               # x = x - 1
16         CALL pthread_exit         # T2 finish

17 <T3>:
18         MOV x, %eax
19         ADD 1, %eax
1A         MOV %eax, x               # x = x + 1
1B         CMP x, 1                  # x == 1?
1C         JE 1E
1D         CALL pthread_exit         # T3 finish
1E         (...)                     # Prepara stack de printf
1F         CALL printf
```

Formatação da Sequência de Execução
A apresentação das sequências de execução das threads que imprimem determinado valor foi formatada segundo pares, da forma:

(thread x, linha x) - (thread y, linha y) - ... (thread z, linha z)

Quando duas linhas são executadas em paralelo, envolveu-se tais pares por um grande parênteses, da forma:

$$\Big((\text{thread } x, \text{ linha } x), (\text{thread } y, \text{ linha } y)\Big)$$

que expressa, nesse caso, que a linha x da thread x e a linha y da thread y foram executadas simultaneamente em processadores distintos.
E por fim, para cada sequência de comandos (em C), mostra-se sua sequência equivalente em *assembly*, tendo como referência, o código acima.

Valor 1 O valor 1 poderia ser impresso na saída padrão se as seguintes threads executarem, nessa ordem, as seguintes linhas:

$$(T2,1) - (T3,1) - (T1,1) - (T3,2) - (T3,3)$$

Essa não é a sequência mais simples de impressão do 1, mas também imprime ele. A sequência mais simples seria se somente $T3$ executasse, sem participação das outras.
Se considerarmos o código de montagem posto acima, o fluxo do programa para imprimir 1 seria:

$$10 - 11 - 12 - 18 - 19 - 1A - 1 - 2 - 3 - 1B - 1C - 1E - 1F$$

Valor -1 O valor -1 poderia ser impresso na saída padrão se as seguintes threads executarem, nessa ordem, as seguintes linhas:

$$(T1,1) - (T1,2) - (T1,3) - (T1,4) - (T1,5)$$

Essa é a execução de $T1$, do início ao fim. Assim, o código de montagem de -1 seria:

$$1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - A - B - D - E$$

Valor 0 O valor 0 poderia ser impresso a partir da seguinte sequência de execução:

$$(T1,1) - (T1,2) - (T1,3) - (T1,4) - (T2,1) - (T1,5)$$

O $T1$ executaria do início até a verificação (CMP) do if e antes que $T1$ começasse a preparar a pilha de impressão do $printf$, $T2$ executaria, modificando o x e fazendo o ser 0. Quando o controle voltasse para $T1$, ele empilharia o valor de x, que agora é 0, devido a modificação de $T2$, e portanto, tal valor seria impresso na saída padrão.

Considerando o assembly, tal sequência se traduz:

$$1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - A - 10 - 11 - 12 - B - D - E$$

Valor 2 O valor 2 poderia ser impresso a partir da seguinte sequência de execução:

$$(T3,1) - (T3,2) - (T2,1) - (T3,3)$$

Como no caso anterior (de impressão do 0), $T3$ faria sua execução do início até a verificação (CMP) do if e antes de empilhar o x para chamar $printf$, $T2$ executaria, modificando o x e fazendo o ser 2, que então quando o controle voltasse para $T3$, $x = 2$ é o que seria empilhado e, portanto, é o que seria impresso

Em assembly, tal sequência se traduz:

$$18 - 19 - 1A - 1B - 10 - 11 - 12 - 1C - 1E - 1F$$

Valor -2 O valor -2 talvez pudesse ser impresso, a partir da seguinte sequência de instruções, em que supõe-se que há, pelo menos, três processadores na máquina e que esses executam em paralelo certas threads.

$$(T1,1) - \underbrace{\Big((T1,2),(T2,1),(T3,1)\Big)}_{\text{paralelo}} - (T1,3) - (T1,4) - (T2,2) - (T1,5)$$

$T1$ começaria executando e então, três processadores distintos escalonariam, respectivamente, $\Big((T1,2),(T2,1),(T3,1)\Big)$. Nesse momento, cada processador computa $x = 0$, já que todos recebem $x = -1$ de $(T1,1)$ e todos adicionam 1 a x. Assim, todos escrevem $x = 0$ na memória, que é o valor que segue para o restante da computação das threads, que imprimem o -2, de acordo com o restante da sequência.

Em assembly, tal sequência se traduz:

$$1 - 2 - 3 - \begin{cases} 4 - 5 - 6 \\ 10 - 11 - 12 \\ 18 - 19 - 1A \end{cases} - 7 - 8 - 9 - A - 13 - 14 - 15 - B - D - E$$

Valor 3 O valor 3 talvez pudesse ser impresso, se dois processadores distintos escalonarem duas linhas de threads distintas com expressões aritméticas distintas para x, mas a escrita em memória de uma delas, que prevalecesse. Assim, uma sobreescreveu a outra.

$$(T3,1) - (T3,2) - \underbrace{\Big((T1,1),(T2,1)\Big)}_{\text{resulta em } (T2,1)} - (T1,2) - (T3,3)$$

$T3$ executaria até o CMP do if e em sequência, dois processadores distintos executariam, em paralelo, respectivamente, $\Big((T1,1),(T2,1)\Big)$

Ao final das operações aritméticas dessa execução em paralelo, cada processador terá um valor distinto para x. Um processador terá $x = 0$ (resultado de $(T1,1)$ na sequência) e outro terá $x = 2$ (resultado de $(T2,1)$ na sequência). E assim, quando fossem escrever o valor de x na memória, como tal escrita é enfileirada, pelo fato da arquitetura moderna dos computadores serem de vários processadores que compartilham a mesma RAM, que possui apenas um controlador, um desses valores de x prevalecerá. Se $x = 2$ fosse tal valor, então o restante da sequência é capaz de imprimir 3.

O código de montagem do que foi dito, torna mais clara essa explicação:

$$18 - 19 - 1A - 1B - \begin{cases} 1-2 \\ 10-11 \end{cases} - \underline{3} - \underline{12} - 4 - 5 - 6 - 1C - 1E - 1F$$

Na sequência $\underline{3}$ é a linha em assembly que $T1$ escreve seu resultado na memória. E $\underline{12}$ é a linha em assembly que $T2$ escreve seu resultado na memória. Como essa escrita acontece de forma sequencial, necessariamente, devido a arquitetura dos computadores modernos, o valor 3 pode ser impresso se a execução em assembly seguir necessariamente essa ordem. Onde $\underline{12}$ sobreescreve o valor de x, posto por $\underline{3}$. Ou seja, a $(T2,1)$ sobreescreve $(T1,1)$ na memória.

Valor -3 O valor -3 não é capaz de ser impresso. Podemos justificar isso da seguinte forma: Só há duas maneiras de imprimir o valor de x. Uma via $T1$ e outra via $T3$. Se começarmos por $T3$, estaríamos dificultando o fato de x se tornar -3 durante a computação dos restantes das threads. Na realidade, se começarmos executando por $T3$, visando imprimir o x, não há sequência possível que produza $x = -3$, na memória. O maior valor atingido é $x = -2$, mesmo considerando casos de execução simultânea de instruções, por processadores distintos, em que há sobreescrita de informação na memória.

A sequência abaixo, mostra a obtenção do $x = -2$, maior acúmulo possível, se começarmos por $T3$.

$$(T3,1) - (T3,2) - \underbrace{\Big((T1,1),(T2,1)\Big)}_{\text{resulta em (T1,1)}} - \underbrace{\Big((T1,2),(T2,2)\Big)}_{\text{resulta em (T2,2)}} - (T1,3) - (T3,3)$$

Em assembly, tal sequência se traduz:

$$18 - 19 - 1A - 1B - \begin{cases} 1-2 \\ 10-11 \end{cases} - \underline{12} - \underline{3} - \begin{cases} 4-5 \\ 13-14 \end{cases} - \underline{6} - \underline{15} - 7 - 8 - 9 - 1C - 1E - 1F$$

Por outro lado, se começarmos por $T1$, conseguimos acumular $x = -3$ na memória, porém não conseguimos imprimí-lo, pois ele não passa nas condições do if. A sequência abaixo expressa isso:

$$\underbrace{\Big((T1,1),(T2,1)\Big)}_{\text{resulta em (T1,1)}} - \underbrace{\Big((T1,2),(T2,2)\Big)}_{\text{resulta em (T2,2)}} - (T1,3) - \ ??$$

Ao final da instrução $(T1,3)$ temos $x = -3$ na memória, mas não há como imprimí-lo. Assim, -3 é um valor que não pode ser impresso. Em assembly, a tradução é mostrada abaixo:

$$\begin{cases} 1-2 \\ 10-11 \end{cases} - \underline{12} - \underline{3} - \begin{cases} 4-5 \\ 13-14 \end{cases} - \underline{6} - \underline{15} - 7 - 8 - 9 - \ ??$$

Valor 4, -4 Ambos os valores $4, -4$ não são capazes de serem impressos. Como só há três threads, que conseguem incrementar ou decrementar uma unidade ao valor de x, é impossível obter um acúmulo para x, tal que $|x| > 3$. Tal acúmulo só seria possível se considerarmos processadores distintos escalonando a mesma linha da mesma thread, mas isso é definitivamente absurdo, logo tais valores não podem ser impressos ou sequer obtidos na memória.